Proceedings of the 1992 IEEE
International Conference on Robotics and Automation
Nice, France - May 1992

Modular Approach for Petri-Net Modeling of Flexible Manufacturing Systems Adaptable to Various Task-Flow Requirement

Chin-Jung Tsai & Li-Chen Fu

Department of Computer Science & Information Engineering
National Taiwan University, Taipei, Taiwan, R.O.C.

Abstract

In this paper, a modular approach is presented for construcing Petri-net models for a class of flexible manufacturing systems (FMS's) composed of a transportation vehicle and several functional groups of entities such as machines and buffers. The resulting model preserves the geometric characteristics of the transportation subsystem as well as the flexibility of alternative routes for material flow in an FMS. By separating machine-dependent part from the whole system, the final model in modular structure is adatable to various task flow requirements. In addition, the methodology can deal conveniently with reconfiguration of transportation layout.

1 Introduction

Petri net has been applied to various aspects of factory automation and extended to flexible manufacturing systems (FMS's), such as modeling, analysis, evaluation, simulation, controller design, etc.[1,2,3,5, 6,11,12,14,15,16]. Apparently, to run a general system successfully the first thing to begin with is to establish a suitable model which can reflect the behavior of the system as much as possible. An FMS is a large complex system consisting of many shared resources connected by the transportation subsystem so that considerable flexibility arises from existence of alternative routes for the material flow. Unfortunately, this also causes great amount of complexity in the work of modeling.

The work of this paper is to propose a buttom-up modular approach, rather than a top-down one [16], and to introduce the concept of separation between machine-dependent module and machine-independent module. Such modeling approach adopted here presents a hierarchical model featured in its systematically defined modular structure in comparison with the earlier works [6,12,14].

This paper addresses the modeling of a class of FMS's comprising one transportaion vehicle and several functional groups of entities such as machines and buffers. Here, we propose a systematic approach of constructing Petri-net model for an FMS mentioned above module by module so that the geometric characteristics of the transportation subsystem and the flexibility of the material flow with alternative routes can be truthfully reflected.

In such a modular approach, two major parts that constitute the whole system are separated, one modeled as a stationary module dependent on transportation layout whereas the other modeled as a variable module dependent on task-flow requirements. Because of the modular structure of the resulting model, the adaptability of this approach to various task-flow requirements is clear.

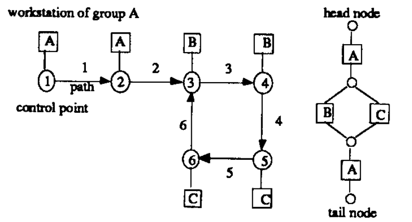
The organization of the paper is given as follows: In section 2, we formulate the problem to be addressed. In section 3, we present our modular approach by giving a series of module developments in terms of an example. A relavant discussion and a conclusion are provided in section 4.

2 Problem Formulation

In this paper, the target system under our modeling consideration can be described in three separate parts, i.e. transportation layout, workstation group definitions, and task-flow specification. Physically, a transportation layout comprises a set of control points and a set of paths, where each path connects a pair of control points, and thereby the vehicle transports materials from one control point to another. Here, for simplicity we will consider a system which only has one material handling vehicle that can move forward and backward on each path. Thus, if control points and paths are viewed as nodes and unidirected edges (arcs) repectively, then the transportation layout can be uniquely represented by a (node-arc) incidence matrix $A = [a_{ij}]$ associated with the unidirected graph $G = (V, E)$, where V is the set of nodes to represent control points, E is the set of arcs to represent paths, and

$$a_{ij} = \begin{cases} +1 & \text{if arc } e_j \text{ leaves node } v_i \\ -1 & \text{if arc } e_j \text{ enters node } v_i \\ 0 & \text{otherwise} \end{cases}$$

Each control points is attached to an entity in the system, called workstation hereafter, which can be a machine, a buffer, a loading/unloading entry, etc. These workstations are then classified into several groups according to what operations they per-



(a) Transportation Layout (b) Task Flow Requirement

Figure1 An Example of Problem Formulation

form. The workstations in the same group can provide the same set of operations, but probably with different parameters, e.g. processing time. To specify such group relation, we use a relation matrix $R = [r_{ij}]$ defined as:

$$r_{ij} = \begin{cases} 1 & \text{if workstation at node } v_i \text{ belongs to group } j \\ 0 & \text{otherwise} \end{cases}$$

For simplifying the matters, throughout this paper we will assume that all workstations in the same group are equally preferable whenever they are available. In other words, for any two free workstations in a group, no priority will be set in drafting any of the two.

According to the definition of the group relation, we can now give the formal description of any task-flow requirement by an AND/OR graph with only OR nodes, as shown in Fig. 1(b), where the head node and the tail node indicate start and end of the task flow respectively. In fact, any general AND/OR graph can be redraw into the type of graphs as just mentioned. Each path in the task-flow graph connecting the head node and the tail node represents an alternative route through various workstation groups, rather than individual workstations.

To illustrate the target system under our problem formulation, an example is shown in Fig. 1, where the transportation layout incidence matrix A and the group relation matrix R are given as follows:

$$A = \begin{bmatrix} 1 & & & & & \\ -1 & 1 & & & & \\ & -1 & 1 & & & -1 \\ & & -1 & 1 & & \\ & & & -1 & 1 & \\ & & & & -1 & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} 1 & & \\ 1 & & \\ & 1 & \\ & 1 & \\ & & 1 \\ & & 1 \end{bmatrix}$$

3 Modular Approach in Modeling

In constructing a complete model to reflect the behavior of an FMS, hardware configurarion and various task-flow requirements are two essential parts to form the machine-dependent and machine-independent characteristics of the system repectively. The key point of this paper is to model these two parts in separate modules, i.e. a stationary module which is only based on geometric features of the pertaining transportaion layout and a variable module which is based on various task-type specification. Due to this separation, the resulting model is easily adaptable to a change of any task-flow requirement because no efforts need to be made to modify the part of machine-dependent module. So far, these two modules, in fact, play the roles of macro-modules since each of them is further modeled in a structure with micro-modules that is naturally formed through our problem description.

Note that capability of the transportation subsystem is the primary function of the machine-dependent module. Thus, to model this part the Transportation Module is carefully devised, i.e. its objective is to model vehicle movement from the current control point to the desired one according to some embedded rules for receiving moving commands, executing the moving activities, and sending outside acknowledgement after each activity is complete. The machine-independent module called Task Flow Module keeps the capability of assigning alternative route of controlling the flow of a workpiece. Between the two modules mentioned above, we need another module, namely, Command Control Module, to execute the command of moving from the current control point to the destination control point. Hence, the main body of one moving command comprises start of command, movement to the current control point, release of workstation at the current control point, movement to the destination control point, replacement of the current control point with the destination control point, and end of command. The details of these modules will be discussed in the following.

3.1 Transportation Module

Given the directed graph $G = (V, E)$ representing the transportation layout, we can define a movement control matrix $M = [m_{ij}], 1 \le i, j \le |V|$ and $|m_{ij}| \le |E|$ to indicate the first directed edge that will be transversed by the movement from v_i to v_j, i.e.

$$m_{ij} = \begin{cases} +k & \text{if on path segment } e_k \text{ in positive direction} \\ -k & \text{if on path segment } e_k \text{ in negative direction} \\ 0 & \text{if } i = j \end{cases}$$

Next, the Petri-net model of the Transportation Module can be systematically established according to the incidence matrix and the movement control matrix as shown in Example 1. Likewise, with different configuration of transportation layout the same approach as adopted in this example can be applied to construct the suitable Petri-net model in a similar form.

Example 1 In this example, we consider an FMS comprising three different machines, one buffer pool of three entities, and one loading/unloading entry, as pictured in Fig. 2, where control points v_1 is located

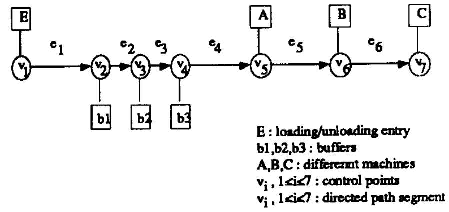


Figure 2 An Example of Transportation Layout

at the loading/unloading entry, v_2, v_3, and v_4 are assoicated with three buffers of a group, and v_5, v_6, and v_7 are attached to machine A, B, and C respectively with different capability. The incidence matrix A associated with this transportation layout is

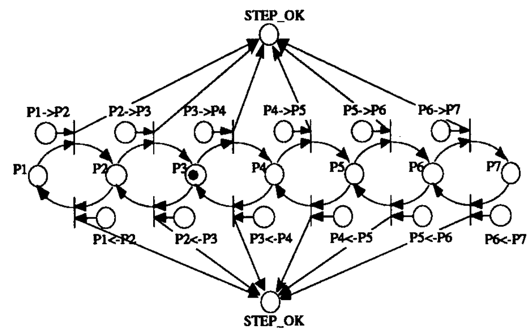
$$A = \begin{bmatrix} 1 & & & & & \\ -1 & 1 & & & & \\ & -1 & 1 & & & \\ & & -1 & 1 & & \\ & & & -1 & 1 & \\ & & & & -1 & 1 \\ & & & & & -1 \end{bmatrix}$$

whereas the movement control matrix is defined in a straightforward manner as:

$$M = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & 0 & 2 & 2 & 2 & 2 & 2 \\ -2 & -2 & 0 & 3 & 3 & 3 & 3 \\ -3 & -3 & -3 & 0 & 4 & 4 & 4 \\ -4 & -4 & -4 & -4 & 0 & 5 & 5 \\ -5 & -5 & -5 & -5 & -5 & 0 & 6 \\ -6 & -6 & -6 & -6 & -6 & -6 & 0 \end{bmatrix}$$

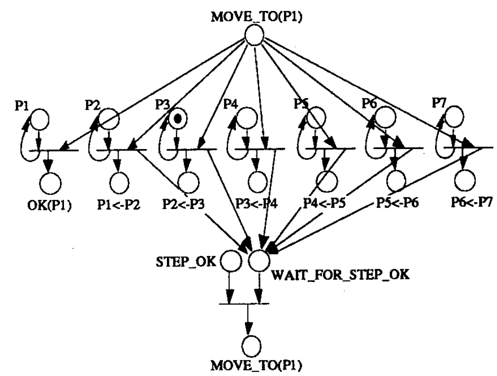
Accordingly, the Petri-net model of this Transportation Module is presented in Fig. 3, where part (a) is related to the geometric layout of the transportation subsystem and part (b) relates what is shown on the first column of the movement control matrix M. An observation of Fig. 3 shows that, after the place $MOVE\_TO(P1)$ receives a token from outside (specificially from Command Control Module to be discussed shortly), a movement to place $P1$ is enforced. Thus, the token representing the transportation vehicle, which initially appears in a place (here is $P3$) associated with a physical starting control node (here is v_3), will begin to move until the destination place $P1$ is reached, associated with another physical terminating control node v_1. Finally, the token which appears in the place $OK(P1)$, is used as the ackownledgement of completion of movement.

We should note that part(b) itself is a submodule (or micro-module) which handles the movement control from any initial control node to the control node v_1. Actually, there are six other similar submodules that take care of various movements to other terminating control nodes, namely v_2 to v_7.



● : a token stands for a transportation vehicle

(a) Petri-net Model Dependent on Transportation Layout



(b) Petri-net Model Dependent on Transportation Control

Figure 3 Petri-net Model of the Transportation Module

3.2 Task Flow Module

To begin with, we assign each node in a task-flow graph an identity number j. Additionally, each workpiece is given an associated Task Flow Module with an index i, where each module is again composed of a number of submodules, each called Task Unit Module as shown in Fig. 4, in the same structure as that of the corresponding task-flow graph. Each submodule just mentioned is given a module name $TASK\_UNIT(i, j, T, P1, P2, ..., PN)$, where i is the indexing number of workpiece, j is the identity number indexing the node in the task-flow graph, and T is the type name of the corresponding workstation group which has N workstations denoted as $P1, P2, ...PN$, respectively.

The notations in Fig. 4 are explained as follows. The place $GROUP\_AVL(T)$ is initialized with N tokens to indicate the number of the available worksta-

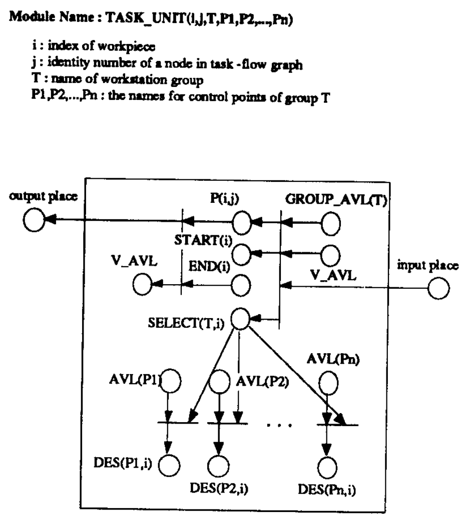


Figure 4 The Task Unit Module

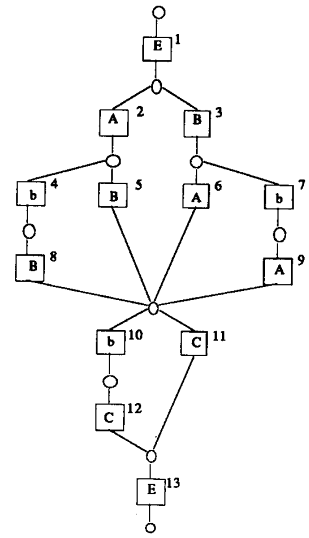


Figure 5 The Task-flow Graph of Example 2

tions in group T. The place V_AVL has a token to indicate that the transportation vehicle is currently available. The places $AVL()$ are used to hold the availability status of workstations in the group T. The token in place $SELECT(T,i)$ will choose an available workstation of the group T to be a new destination, i.e. a token will appear in one of the places $DES(,i)$. The token in place $START(i)$ then activates the movement command in Command Control Module with the destination information being started in places $DES(,i)$. After the completion of the movement command, a token takes place in $END(i)$ to send out ackowndlegement of this completion message. Note that the place $P(i,j)$ in the submodule demonstrates in Fig. 4 has the function of identifying which Task Unit Module is currently activated.

The functional description of the Task Unit Module is stated as follows. Once a permissible workstation group and the transportation vehicle are both available, the token in the input place enters the module right away to select an entity of the group as the new destination and to start the activity in the Command Control Module. It will be clear later that at the end of such activity, the transportation vehicle finishs the movement from the current place to the destination place and then make replacement of the current place with the destination place. After that, a token returns to the place V_AVL to indicate the release of the transportation vehicle and incidentally the output place obtains a token to inform the outside that the subtask handled by this submodule is completed. As to release of the workstation, it is included in the Command Control Module as will be shown subsequently.

It is noteworthy that the structure of the Task Flow Module is the same as that of the task-flow graph, which allows the flexibility of alternative routes and the control of the task flow both to be established through this modular approach. We illustrate this feature through the modular combination in Example 2.

Example 2 With the same hardware configuration in Example 1, we define the desired task type in the task-flow graph as shown in Fig. 5, where each node is assigned an identity number. Thus, the final form of the Task Flow Module is constructed through modular combination as shown in Fig. 6.

3.3 Command Control Module

Between the Transportation Module and the Task Flow Module, we need another module called Command Control Module whose functions consist of the following:

i) to register the current control point at which the transportation vehicle rests,

ii) to start the activity of movement from the current control point to the destination control point according to the task flow in the Task Flow Module,

iii) to cause the vehicle to move to the current control point in Transportation Module,

iv) to release the workstation associated with the current control point,

v) to initiate another movement of the vehicle to the destination control point following the Task Flow Module,

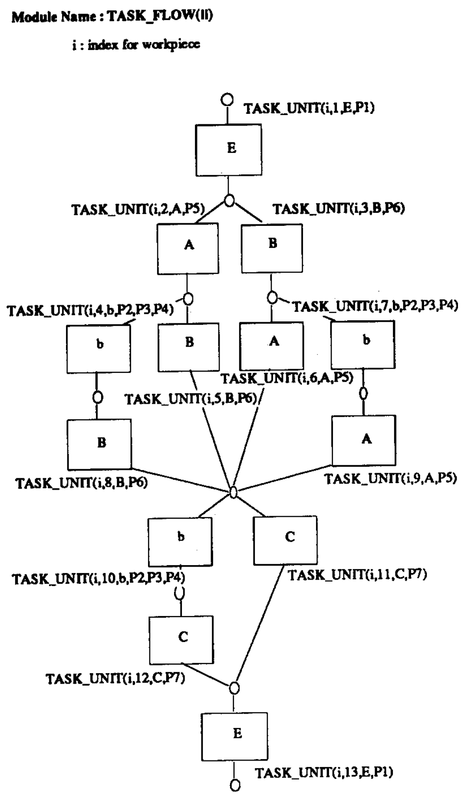


Figure 6 The Task Flow Module of Example 2

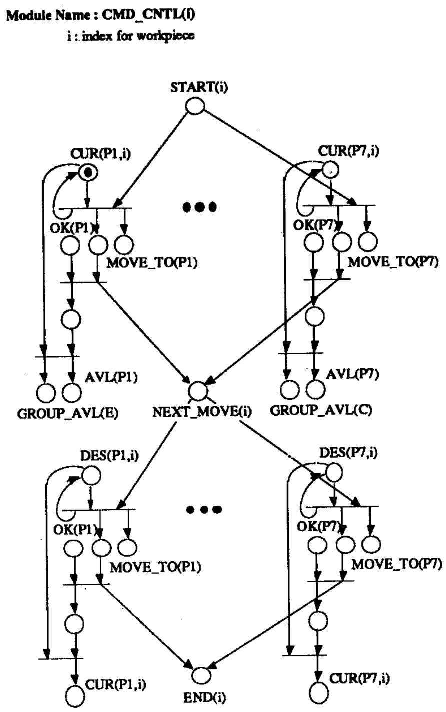


Figure 7 The Command Control Module of Example 3

vi) to replace the current control point with the destination control point,

vii) to end the activity mentioned above.

As for the Task Flow Module discussed in the previous subsection, each module here is assigned a number i to indicate which workpiece is in process. To illustrate the details of this module, an example is given as follows.

Example 3 Considering the same system configuration as in Example 1, then the outline of the associated Command Control Module is pictured in Fig. 7. A brief description of Fig. 7 is given below. The places $START(i)$ and $END(i)$ indicate the status related to start and end of the movement command respectively according to where the token appears. There is always a token in one of the places $CUR(,i)$ to indicate the current control point of the ith workpiece. After the movement to the current place, the place $NEXT\_MOVE(i)$ will enable the next movement to the destination place. At the same time, a token is added to the place $GROUP\_AVL()$ place and the current workstation is released for availability by returning a token to the corresponding place $AVL()$. After the movement to the destination place, the toke in the destination place will move to the corresponding $CUR(,i)$ place to update the location of the current control point.

3.4 A Complete Model

In addition to the characteristics discussed above, in fact, other considerations are needed to describe the complete the behavior of an FMS, such as the freedom from deadlock, efficiency of dispatching rule, etc. But in this work, for the time being details as how to make a successful system model are omitted and only the concept of a modular approach is emphasized and presented. An example of a complete system model is provided in Example 4.

Example 4 Considering the configuration in Example 1, we can set the limit on the maximum work-in-process to be three to guarantee that the system will never run into a deadlock. The final completed model is in Fig. 8, where the tokens in $WORK()$ places are used to separate the flow of each workpiece and three

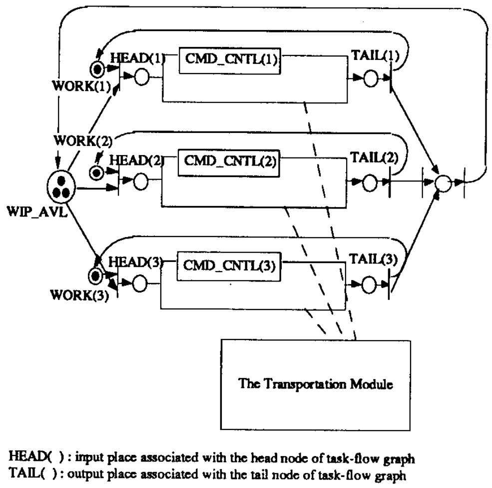


Figure 8 The Outline of the system Model in Example 4

tokens in place WIP_AVL set the maximum work-in-process.

4 Discussions and Conclusions

Following the concept of the approach presented in this paper, we can conveniently extend the modeling methodology to the case of multiple task -types in process. Although no direct formation related to performance evaluation is presented in this work, the time factors can be easily added to the Petri-net model of each module. After that, we can develop futher work about performance evaluation by using simulation techniques or other tools based on Petri-net models.[1,3,4,6,7,8,9,10,13]

While considering a more complicated FMS than a simple one, such as with multiple material hadling vehicles, complex layout of the transportation system, etc., the problem of deadlock becomes more difficult to deal with. At the same time, how to build efficient rules of dispatching among alternatives also needs to be solved. To achieve a complete model of a general FMS, more efforts are needed to contribute to resolution of these open problems, e.g. a general for rule solving deadlock problem for certain kind of FMS, a methodology for combination of modules and rules to construct a complete system model, an efficient dispatching rule for some system configuration, etc.

This paper proposed a new concept using Petri nets to model flexible manufacturing system in modular approach. Its main features are to separate the task-dependent part from the complete model of the system, to build the system model in modular structure, and to consider geometric characteristics of transportation system as one essential element of the whole system.

Such an approach is easily adaptable to a change of any of the task-flow requirements and different layout of the same kind of the transportation system. From this viewpoint, it provides a systematic methodology to generate a prototype system model according to a formal specification of an FMS. Ongoing research will be to integrate the prototype model with any suitable control modules containing more physical rules to run the whole system, or to tie it with some Petri-net-based techniques to perform analysis, evaluation, simulation, etc.

References

[1] A. D. Bruno and M. Morisio, "Petri-net based simulation of Manufacturing cells," *Proc. IEEE Robotics Automat. Conf.* 1987 , pp. 1174-1179.

[2] D. Crockett, A. Desrochers, F. DiCesare, and T. Ward, "Implementation of a Petri net controller for a machining workstation," *Proc. IEEE Robotics Automat. Conf.* (Raleigh, NC), Mar. 1987, pp. 1861-1867.

[3] M. A. Holliday and M. K. Vernon, "A generalized timed Petri net model for performance analysis," *IEEE Trans. Software Eng.*, vol. SE-13, pp. 1297-1310, 1987.

[4] K. Jensen, "Coloured Petri nets and the invariant method," *Theoretical Computer Science*, vol. 14, pp. 317-336, 1984.

[5] M. Kamath and N. Viswanadham, "Applications of Petri net based models in the modeling and asnaysis of flexible manufacturing systems," *Proc. IEEE Robotics Automat. Conf.* April 1986, pp. 312-316.

[6] J. Martinez, P. Muro and M. Silva, "Modelin, validation and software implementaion of production systems using high level Petri nets," in *Proc. IEEE Robotics Automat. Conf.* (Raleigh, NC) April 1987, pp. 1180-1185.

[7] M. K. Molly, "Performance analysis using stochastic Petri nets," *IEEE Trans. Comput.*, vol. C-31, pp. 913-917, 1982.

[8] M. K. Molly, "Discrete time stochastic Petri nets," *IEEE Trans. Software Eng.*, vol. SE-11, pp. 417-423, 1985.

[9] T. Murata, "Synthesis of decision-free concurrent systems for prescribed resources and performance," *IEEE Trans. Software Eng.*, vol. SE-6, no. 6, pp. 525-530, 1980.

[10] T. Murata,"Petri nets: properties, analysis and application," *Proc. IEEE*, vol. 77, no. 4, pp. 541-579, 1989.

[11] T. Murata, N. Komoda, and K. Matsumoto, "A Petri net based controller for flexible and maintainable sequence control and its application in factory automation," *IEEE Trans. Ind. Electron.*, vol. IE-33,, pp. 1-8, 1986.

[12] Y. Narahari and N. Vismanadham, "A Petri net approach to the modeling and analysis of flexible manufacturing systems," *Ann. Operations Res.*, vol. 3,, pp. 449-472, 1985.

[13] C. V. Ramamoorthy and G. S. Ho, "Performance evaluation of asynchronous concurrent systems using Petri nets," *IEEE Trans. Software Eng.*, vol. SE-6, pp. 440-449, 1980.

[14] N. Viswanadham and Y. Narahari, "Coloured Petri net models for automated manufacturing systems," in *Proc. Robotics Automat Conf.* (Raleigh, NC) April 1987, pp. 1005-1011.

[15] N. Zerhouni and H. Alla, "Dynamic analysis of manufacturing systems using continuous Petri nets," *Proc. IEEE Robotics Automat. Conf.* 1990, pp. 1070-1075.

[16] M. C. Zhou, F. DiCesare, and A. A. Desrochers, "A top-down modular approach to synthesis of Petri net models for manufacturing systems," *Proc. IEEE Robotics Automat. Conf.* (Scottsdale, AZ), 1989, pp. 534-539.